Swathbuckler: Real-Time Wide Swath Synthetic Aperture Radar Image Formation Using Embedded HPC

Richard W. Linderman and Joshua Corner
US Air Force Research Laboratory/Information Directorate (AFRL/IF), Rome, NY
{Richard.Linderman, Joshua.Corner}@rl.af.mil

Scot Tucker
Advanced Engineering & Sciences, ITT Industries, Rome, NY
Scot.Tucker@rl.af.mil

Abstract

This high performance computing (HPC) system processed real-time X-band radar returns into continuous strip-map, high-resolution (< 1m), wide-swath (37 km) imagery on-board a Convair 580 aircraft. The HPC system cost under $100K. The synthetic aperture radar (SAR) image formation algorithm was optimized to achieve real-time processing 9000 times faster than the original algorithm specification. An information management system allowed real-time off-board exploitation of the on-board multi-terabyte database. Real-time SAR image formation was demonstrated on five flight tests. The completed flights provided 7.3 terabytes of raw and processed imagery.

1. Introduction

The Swathbuckler experiment was conducted between 2001 and 2005 under the auspices of The Technical Cooperation Program (TTCP). The goal was to design, implement, and flight test novel system architecture able to continually process in real-time 37 km wide radar returns into high resolution stripmap synthetic aperture radar (SAR) imagery and make this voluminous information product available everywhere within seconds. The US, UK, Canada, and Australia all contributed to the experiment, with the US delivering the embedded HPC with optimized software. The high performance computing (HPC) challenge was to reduce the 12.5 hour runtime of the initial Canadian MATLAB code to two minutes on a single node. The resulting test image was 1.5 km by 11.2 km. Parallelizing across 24 nodes then yielded the 37 km deep image—a speedup of over 9000X—which produced imagery at a rate of 3.43 km^2/second or 300K km^2/day.

As background, stripmap SAR imagery is produced by processing the returns from thousands of pulses emitted by airborne radar as it nominally flies a straight line. The distance flown as these pulses are emitted creates a "synthetic" aperture which is much larger than the radar antenna physically resident on an aircraft. This allows much higher resolution imagery to be created in the direction parallel to the aircraft flight path. High speed A/D converters provide high resolution in the range direction, which runs perpendicular to the flight path. In this case, the pixel sizes were less than 1 meter (see Figure 1 for a sample image).

SAR applications typically involve sophisticated processing of large volumes of data[1] to correct for aircraft motion away from the planned path, compress the signal returns in the range (perpendicular to aircraft) and azimuth (parallel to aircraft) directions, and nicely focus the images. Current SAR systems operate across small range swaths at any one time due to the processing limitations associated with generating the imagery. Here the objective was to parallelize on an embedded HPC to image a 37 km wide range swath in real time. The SAR imagery produced during our demonstration processed returns from blocks of 16384 or 32768 transmitted pulses into images and wrote them to disk.[2]

The HPC computational challenge was primarily related to speeding up the initial MATLAB algorithm to keep up with the incoming pulses as the aircraft flew. However, accepting high speed inputs, and writing finished images to disk were also significant challenges, especially when implementing on a conventional HPC cluster to make the system most affordable.

2. Objectives

The principal objective was to demonstrate in flight tests that affordable (~$100K) embedded HPC was up to the challenge of sustaining computational throughput of 100 GFLOPS and thereby allowing high resolution (<1m) SAR imagery across a 37 km strip to be produced and exploited in real-time. The principal subcomponents of this objective were:

HPCMP Users Group Conference (HPCMP-UGC'06)
0-7695-2797-3/06 $20.00 © 2006 IEEE

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Swathbuckler: Real-Time Wide Swath Synthetic Aperture Radar Image Formation Using Embedded HPC	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Advanced Engineering & Sciences, ITT,Rome,NY**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
2006 IEEE Radar Conference, held in Verona, NY on April 24-27, 2006

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**8**	

A. High speed data input to the HPC from the radar frontend
B. Optimized real-time signal/image processing on the HPC with storage to disk
C. Real-time exploitation and dissemination
D. Integration of the US HPC with the Canadian radar and aircraft systems and the UK A/D and frontend hardware
E. Flight test experimentation and data collection

3. Approach

3.1. Overview of the Approach

The real-time data input used an FPGA to fan-out the high speed A/D inputs into UDP packets sent to the 24 nodes across eight gigabit Ethernet fiber optic links. The optimization began by converting MATLAB to C++ which reduced azimuth block formation to 4050 seconds. Subsequent optimizations including vector libraries and cache optimizations reduced this down to 15 seconds—meeting the real-time deadline. Integration with the other nations began with data exchanges and proceeded to laboratory testing in Ottawa, followed by five flight tests in Ottawa and Kingston, Ontario in Aug–Sep, 2005. During these flights both the raw radar returns and the formed images were saved to disk to support follow-on research. The onboard and offboard exploitation used a publish-subscribe-query information management environment onboard, linked via 300 kbit radio and the internet to a like environment on HPCs at Rome, NY and from there disseminated to users across the DREN.

3.2. Embedded HPC (EHPC) Selection and Design

The evolution of the EHPC system architecture design identified key characteristics of the computing platform that would provide the most benefit to the SAR image formation task. Because stripmap SAR image formation tends to be memory and I/O bound, increases in DRAM densities, on-board cache sizes, and I/O bandwidths directly relate to the achievable performance of the EHPC. From the time of initial concept to flight tests several improvements in memory and I/O price performance greatly influenced the final design. For example, the first EHPC design used a SKY multi-computer with 16 PowerPC processors on 8 boards with only 64 megabytes of memory per PowerPC with Fiberchannel I/O cards (~$1.8M). The second iteration explored 24 dual Xeon nodes supported by 4 GB of DRAM, each with an associated Annapolis Wildstar FPGA board in addition to gigabit and 100 Mb Ethernet links (~$500K). The flight EHPC employed 24 dual Xeon nodes with L2 caches sized at 1MB supported by 8GB of DRAM and dual gigabit Ethernet links (~$75K) with a single FPGA in the frontend to packetize A/D data. This "Coyote" cluster is the EHPC that was used for the Swathbuckler flight tests. More details on the system architecture evolution can be found in Reference 2.

Figure 2 depicts the final system architecture which used the EHPC to form hi-resolution SAR imagery across the 37 km wide swath continuously. The processing load is linearly related to the speed of the aircraft, which for stripmap SAR processing sets the pulse repetition frequency (PRF) of the radar. For this experiment, a PRF of 500 Hz was sustained. With the system architecture established, the design of the subcomponents could proceed.

A. High speed data input to the HPC from the radar frontend

A critical component of the system architecture involved designing a method to move the large data stream from the radar sensor to the EHPC in a compatible format to accomplish the signal processing. Because there were no existing systems to interface the SAR returns to an HPC a solution was designed. The resolution requirements dictated a 2 GHz sample rate. Initially, an Atmel analog-to-digital converter (ADC) evaluation board connected to an Atmel Demultiplexing (DMUX) board were selected and fed into a custom board with a single FPGA and memory. The DMUX slowed the 2 GHz data rate down by a factor of eight to 250 MHz, a frequency which could be handled with careful FPGA design. The Xilinx Virtex II Pro FPGAs had recently become available and supported up to eight high speed serial I/O links, which in turn could drive gigabit Ethernet. The FPGA was assigned several tasks. First, it cut out the range cells of interest from the continuous A/D inputs—this required also receiving a discrete timing pulse from the radar controller to precisely know when each radar pulse was transmitted. Second, since the radar data was sampled at an intermediate frequency (IF), the FPGA applied a standard Hilbert transform and low pass filter to convert it to baseband I,Q conventional representation. Third, the FPGA chopped up the wide 37 km swath into twenty four subswaths of 1.5 km and duplicated the necessary overlapping range bins amongst adjacent subswaths. Finally, the FPGA formed UDP packets addressed to each of the 24 cluster nodes and sent them off via the eight gigabit Ethernet ports. Since the FPGA board was near the radar in the front of the aircraft and the cluster was in the rear of the aircraft, fiber optic Ethernet transducers were used to avoid interference in the noisy RF environment. The FPGA also featured embedded PowerPC cores, which allowed a serial interface to external computers to communicate and

interactively set most of the operating parameters of the design, such as range extents, overlaps, etc.

During testing and integration we discovered that the initial prototype had crosstalk issues on the incoming signals from the DMUX board. The integration issues and cross-talk challenges experienced on the prototype motivated the use of an integrated solution—which became a commercial product. The final flight solution used the Quixilica Neptune and Callisto FPGA-based cards. The Neptune card accomplished the ADC and DMUX functions while the conversion to, and distribution of, multiple Gigabit Ethernet outputs capabilities were accomplished with the Callisto VXS FPGA card. Further detail on the design of the interface between the SAR radar and HPC are found in Reference 5.

B. Optimized real-time signal/image processing on the HPC with storage to disk

Meeting the real-time signal/image processing requirements on an HPC requires intimate familiarity with the processing architecture in conjunction with parallel computing techniques. Each component of the HPC contributes significantly to meeting the real-time SAR imaging challenge from the network I/O throughput to processing performance to applying classes of optimization techniques to disk I/O and the size of on-board/on-chip memory. This section details the influence of these architecture components, the algorithm implementation tradeoffs, and other system tradeoffs that contributed to the optimization of the SAR program.

The stripmap processing is grouped into two categories in alignment with the dual-node architecture: range and azimuth processing. Range processing operates continuously on the incoming range lines and consists of parameters generation, motion compensation, forward FFT/range replica and multiply/inverse FFT. Azimuth processing operates on a block of range lines (i.e., an azimuth block), which are used to form the output image and consists of range cell migration correction, autofocusing, azimuth compression, output resampling, pixel detection, and imagery scaling.

1) Using data decomposition to map the SAR algorithm onto the HPC cluster

To maximize the parallel computing power of the HPC the algorithm implementation decomposed the data according to swath width. The 37 km swath divides easily into 24 pieces such that any one cluster node has the responsibility to form ~1.5 km swath of SAR imagery. This allocation allowed for processing at each node on a localized data set. The large number of FFT calculations that make up the range and azimuth processing can operate on each node's local memory eliminating otherwise costly FFT inter-node communication. In addition, all-to-all communications would be required during multiple corner-turning operations between the range and azimuth processing. Using data decomposition the cluster nodes can devote almost all of their time to processing imagery (in chunks of 16 or 32 K blocks depending on range) using local memory structures rather than having to wait on expensive inter-node communication.

Figure 3 represents this data decomposition relating the processed imagery (16K/32K blocks) to each ~1.5 km subswath along one leg of the flight path.

2) Network I/O

The external network connection resident on each node must be able to receive the 16 MB/s of data coming from the high speed front-end interface. The gigabit Ethernet connections local to each node satisfied this requirement.

3) Processing performance and strategy

Signal and image processing applications use floating point operations extensively. Using a complexity analysis one can estimate the anticipated number of FLOPS required for a solution.

Table 1 shows the contributing algorithm complexities of the range processing. For example, the computational complexity for a 32K FFT is $5n \log(n)$ where $n=32,768$ resulting in 2,457 KFLOPS. The vector multiply, motion compensation, and scaling functions are measured by counting how many FLOPS are attributed to addition/multiplication of the vector and scalar operations. For example a complex vector multiplied by a complex vector is attributed 6 FLOPS, an addition of a real vector and a real scalar 1 FLOP, and the cos of a complex vector 80 FLOPS. The total sustained range processing for one node is 2.59 GFLOPS/second assuming we are collecting 500 range lines a second.

Table 1. Range Processing FLOPS

Range Component	Size	Count	FLOP
Convert	32768	1	32768
32K FFT	32768	1	2457600
Vector Multiply	16384	1	98304
Motion Compensation	16386	1	1441968
16K IFFT	16384	1	1146880
Scaling	10390	1	20780
Total GFLOPS per range line (pulse)			0.0051983
Total GFLOPS per node per second			**2.6**

The various components of the azimuth processing shown in Table 2 indicate that it takes ~78 GFLOPS to process and output a 32K block (about 10000 by 13000 pixels) of imagery—which represents what one node would have to accomplish during the allotted timeframe.

Unlike the range processing, the azimuth processing requirements are more grow with range from the aircraft because the overlap between successive azimuth blocks grows. Table 2 is representative of the 10th node (swath range from 34.6-36.2 KM) and must have its first 32K block of data processed in azimuth 50 seconds before the next block must be processed (node 1 only has 24 seconds and uses 16K block sizes.) The total GFLOPS for node 10 is then 78 GFLOPS accomplished in 50 seconds or 1.56 sustained GFLOPS/second. On the average this number is 1.52 GFLOPS/second for each of the 24 nodes for a grand total of 36.6 GFLOPS/second required azimuth processing.

Table 2. Azimuth Processing Operation Count

Azimuth (Block 32K and Presum 2)

Azimuth Compression	Size	Count	GFLOP
Presum Value	32768	10390	25.53
Range Cell Migration Correction	16384	10308	5.66
Comp-Matched	16384	10308	14.86
Comp-IFFT	16384	10308	11.82
Azimuth Compression subtotal			**57.88**
Autofocus			
Select Lines			
Matched Filter	4096	2600	0.94
FFT	4096	2600	0.94
MapDrift			
Matched nAzPs	16384	2600	3.75
8K IFFTx2 nAzPs	16384	2600	5.96
FFT nAzPs	16384	2600	2.98
IFFT nAzPs*2	32768	2600	6.39
Autofocus subtotal			**20.66**
Grand Total per 32K Block			**78.54**

The total range requirement of 62.38 GFLOPS combined with 36.6 GFLOPS of azimuth/output processing requires sustaining on the order of 100 GFLOPS per second for the stated radar configuration. Because the Coyote cluster has a peak of 614 GFLOPS and the FFT benchmarks looked favorable 100 GFLOPS seemed achievable.

4) Classes of optimization techniques

Unfortunately, achieving 16% of peak performance is not as simple as running with '*gcc -O3*'—the highest compiler optimization flag. In our experience it took a persistent effort at several entry points in the software and hardware architecture. The optimization of the SAR task revealed the strengths and subtle relationships that have a large influence on large data signal processing applications. Six classes are defined and analyzed to reveal the most valuable optimization activities.

At the operating system (OS) level optimizations resulted from modifying the kernel configuration to handle larger memory allocations per process (approaching the entire 32-bit pointer limitation: ~4GB), enabling the shared memory extensions in the kernel configuration, and turning on the DMA hooks to avoid thrashing the cache unnecessarily.

The second class of optimization is parallel techniques commonly used in cluster computing. The SAR algorithm benefited by using a 4 stage pipeline so as to minimize waiting that would have resulted from the data-dependencies laden in strictly sequential processes. A second fundamental benefit was an auto-load balancer which monitored the activity for both the range and azimuth processors to take advantage of the extra CPU cycles in situations when one processor's task was finished sooner than the other.

Next, optimizations came from hardware upgrades. More important than clock rate advances were enlarging the level 2 cache from 512K to 1 MB, and recently to 2 MB. This greatly improved performance on longer transforms, which in turn led to less inefficiency with overlaps. For example, understanding how a 16K FFT reacts to L2 cache misses resulted in a 40% speedup after upgrading to 1 MB. While not increasing the speedup directly, upgrading to 8 GB of DRAM did provide for a much larger optimization "landscape" and simplified the shared memory approach to dual Xeon coordination.

"Process specific" optimizations include modifying compilation flags, and changing the program syntax so that the resulting process is streamlined. Often when setting particular optimization flags previously working code has to be modified to fit the tighter compiler constraints. Also part of this class of optimization is the inclusion of special-purpose libraries—such as the vector signal image processing libraries (VSIPL) used extensively in this algorithm. In addition to VSIPL libraries, CPU optimized (e.g., Intel Performance Primitives) were also be included during compile and linking operation.

Eliminating redundancy and reducing I/O overhead through efficient expressions of logic and consolidation are exampes of optimizations in the domain-specific or algorithm class. For example, moving vector allocations outside loops and consolidating expensive corner turn operations were important for stripmap SAR. Removing redundancy and unneeded copying, even in some of the VSIPL libraries, proved invaluable for the azimuth processing task.

Finally, the memory management class seems to have a ubiquitous presence at all size problems. Initially, the MATLAB code managed its intermediate data though the hard disk—this was quickly removed as a major slowdown—however handling 2-3 GB chunks of memory is not easily done on 32-bit operating systems. Equally

challenging is keeping the cache from thrashing and byte-aligning addresses 128 byte boundaries to be considerate of cache line impacts. An important optimization for range cell migration correction was to recast it to work across whole cache lines before stepping, out-of stride, to the next range cell. This reformulation eliminated a costly cornerturn of the large matrix.

Figure 4 shows the chronological contribution of applying these optimizations and highlights the major advances achieved in the case of the azimuth processing. The software optimizations mainly occurred over a seven month period. For a detailed annotation of the events on this chart see Reference 10.

The azimuth processing presented the most difficult optimization challenge. With this in hand, the four stages of the processing were able to sustain the arrival of 500 range lines per second across the 24 nodes. Compared to the MATLAB consumption of 60000 range lines in 12.5 hours for just 1 node's range extent, this is a speedup of 9000X.

In preparation for a second set of flight tests in July 2006, eight new Xeon nodes with 2 MB L2 cache have been procured. With further optimization of the FFT and cornerturning routines, and by increasing the memory available to the threads via kernel modifications, 32K block azimuth processing is available across the whole swath. Azimuth processing has been accelerated to 22 seconds for the 32K block. When combined with the other computational stages, system PRF now exceeds 700 Hz and is approaching the goal of 800 Hz for these flight tests.

C. Real-time exploitation and dissemination

Producing hi-resolution imagery in real-time on an aircraft results in hundreds of megabytes of data being stored to disk every second. Being able to access this large data repository onboard the aircraft might seem to achievable. But providing exploitable access from an offboard site requires a smart information management strategy.

The Joint Battlespace Infosphere (JBI)[7] provides accessible, scalable, custom-query, low latency, redundant, and intuitive information management capabilities—so it was the logical choice. In addition to those attributes the JBI uses a loosely-coupled publish and subscribe architecture.

During the flight tests, in order for the observer on the ground to watch the events as they occurred in the air the information management system had to relay associated meta-data, particularly rectangles on a map portraying the images just formed. After the flights this meta-data can be used to replay each flight in the exact sequence as it occurred. The database contains the processed imagery, time, day, geo-registered location, leg, and block size associated with all the processed imagery. The ground user also uses the meta-data to produce a custom query into the on-board SAR database. On flight 37, the latency in sending, servicing, and delivering the results of a query took 14 seconds.

D. Integration of the US HPC with the Canadian radar and aircraft systems and the UK A/D and frontend hardware

Moving from the protected lab environment to a real-world flight test integration of multiple high performance finely-tuned hardware systems is challenging. This section details experiences embedding an HPC on-board a Convair 580 and ensuring the HPC, front-end hardware, and radar controllers successfully communicated.

The HPC nodes were installed with the first 12 sitting in in front of the remaining 13 (including head node). The weight of the cluster required that it be positioned near the aircraft center of gravity, approximately 30 feet aft of the radar and frontend processing hardware.

The aircraft systems were designed for power loads from prior test configurations which maxed out at 8kW. To host the HPC onboard, two 115V circuits had to be used—a 15-amp for eight of the processing nodes and the head node and a 20-amp for the remaining 16 back-end nodes.

Another challenge involved the aircraft cooling system. Due to the heat generated by an HPC, ground-based installations use raised floor and specialized cooling to avoid CPU damage from excess ambient temperatures. On the Convair, the temperatures were regulated by channeling four passenger air ducts through hoses down to the back and front sides of either cluster rack.

Initially connecting the front-end hardware to the HPC revealed an issue with the network protocol. Since the approved design document used to implement the front-end network interface did not include all levels of the network model, messages coming from the front-end interface were not being properly addressed and packets were being dropped. At the ARP protocol level MAC IDs were not properly associating with node IP addresses. Operation without dropping packets was successfully achieved by running a script on the HPC cluster that forced the correct ARP protocol associations to be made.

Processing the image returns from the radar began with the raw data being fed to the HPC from the front-end interface—but the aircraft navigation data was also needed from the radar system. Collecting the position information from the radar data acquisition system required extracting data fields from binary formatted record structure. Using the header file descriptions and associated receiver simulator the binary data was easily translated into textual representation. The navigation

information was then distributed across the HPC cluster to each of the 24 processing nodes through a mounted network file system.

A successful end-to-end integration test was accomplished using the Range Filter Generation (RFG) mode tests of the radar system. That mode exercised all the components of the radar system, front-end processing system, and the SAR imaging algorithm running on the HPC. The visual representation of the pulse compressed RFG signal in the time domain and frequency domains was used during our final integration tests and in-between flights to identify and resolve unexpected processing results and data representations. It was this RFG test that provided a confidence test before and after each flight leg.

E. Flight test experimentation and data collection

Designing the flight test came out of several meetings of TTCP Sensors Technical Panel on Signal/Image Processing.[3] Hexagonal flight patterns were chosen to image and record six differing angles on the same ground location. These datasets will support subsequent research into fusing multi-perspective imagery into three-dimensional (3-D) models of objects in the scenes. The second goal was to collect imagery around 1) a densely populated city environment and 2) an area centered on a high traffic maritime location. The chosen locations to satisfy those requirements are Ottawa and Kingston, Ontario, both located on the southeast region of Canada. A sample flight path and associated hexagon coverage is shown in Figure 5.

The number of hexagons per flight was limited by the available disk storage space. Initial estimates of 4.8 terabytes of raw and processed imagery per hexagon limited each flight to two hexagons. Six planned flights were to be spread across a three week window with the following pattern: fly on day one; do post processing/analysis for days two, three; and then fly again on day four.

To achieve a data link connection from the aircraft to Rome, NY several options were considered. A 900MHz commercial radio LAN Bridge[4] was the quickest available selection and provided up to 300 Kbits/sec link capacity.

The primary data set is the raw SAR returns. When the SAR radio frequency returns hit the ground area of interest the reflections were collected at an average rate of 500 pulses per second. All digitized samples were distributed across three groups of eight nodes each on the Coyote cluster. This data allows for re-processing the hexagon of information with specifically tuned parameters to get a custom-focused picture.

Although the five flights indicate seven hexagons (42 legs) of data were flown--only a subset of that data was collected due to various problems which eliminated some legs. A total of 3.2 terabytes of raw data was collected. That equates to six legs of data in the Kingston area and 23 legs of data in the Ottawa region.

Processed imagery is the second major data set. A total of 4.1 terabytes of processed data was collected for the same location and number of legs as mentioned for the raw data.

4. Results

The frontend hardware was able to acquire a 37 km swath of 15 cm range cells at a 500 Hz rate and form up 24 overlapped range extents. These were then segmented into UDP packets and sent over 8 gigabit Ethernet links to the 24 processing nodes of the HPC. The general utility of capturing high speed A/D information and packetizing it for consumption by Ethernet based clusters led this solution to become a commercial product.[3]

Work to optimize the real-time SAR image formation processing succeeded in meeting the demands of a 500 Hz PRF system configuration. The overall speedup was approximately 9000X over the initial algorithm specification. Major speedups resulted from use of optimized libraries, eliminating time consuming cornerturn operations, and selecting sizings mindful of Level 2 cache capacities.

On the last flight, the stretch goal of linking the onboard information management environment to a similar environment on the ground via radio link and the internet was accomplished. Metadata published onboard the airplane in Canada appeared on user consoles in Rome, NY within two seconds, and images queried by ground users were served up by the airborne database and returned to the ground within 14 seconds.

After extensive system integration in August 2005, five flight tests were accomplished by mid-September 2005. Hexagonal flight path with 40 km sides were flown with the radar imaging the inside of the hexagon to allow multiperspective looks at the area. In addition to the real-time imaging and exploitation, the raw data was written to disk, as were the images produced. Overall 7.3 TB of data was collected to support future research.

5. Conclusion

Wide swath, high resolution SAR image formation has been a challenging signal/image processing problem to confront HPC computer architectures. However, through a combination of optimization and parallelization, a 9000X speedup has been achieved, which now allows real-time formation of 37 km wide strips of imagery with <1m resolution. Twenty-four dual Xeon nodes costing less than $100K are capable of sustaining the real-time throughput of 100 GFLOPS and continuously produce

imagery at a rate of 3.43 km^2/second. FPGAs allow commercial high speed A/D converters to be connected to the cluster across gigabit Ethernet links. An onboard information management system allows the imagery to be published and exploited remotely within seconds. The system has be successfully flight tested, with a large set of raw radar returns and processed images stored to support future investigations of new HPC architectures and signal/image processing algorithms.

References

1. Soumekh, M., *Synthetic Aperture Radar Singnal Processing with MATLAB Algorithms*, John Wiley & Sons, New York, NY, 1999.

2. Linderman, R., "Swathbuckler: Wide Swath SAR System Architecture." *Proceedings of the 2006 IEEE Radar Conference*, pp. 465–470, April 2006.

3. The Technical Cooperation Program, Sensors Group, http://www.dtic.mil/ttcp/sen.htm, accessed September 2005.

4. Puschel, M. et al., "Spiral: Code Generation for DSP Transforms." *Proceedings of the IEEE special issue on Program Generation, Optimization, and Adaptation*, Vol. 93, No. 2, Feb. 2005.

5. Rouse, S. and D. Bosworth, "Swathbuckler Wide Area SAR Processing Front End." *Proceedings of the 2006 IEEE Radar Conference*, pp. 673–678, April 2006.

6. Damini, A., C. Parry, and G.E. Haslam, "Swathbuckler—Radar System and Signal Processing." *Proceedings of the 2006 IEEE Radar Conference*, pp. 30–34, April 2006.

7. Linderman, R., M. Linderman, and C.S. Lin, "FPGA Acceleration of Information Management Services." *IEEE Military Applications of Programmble Logic Devices Conference*, September 2005.

8. CodeSourcery, LLC, VSIPL++ Specification: 1.0 candidate rev C, 2005.

9. Microhard Systems Inc., *SpectraNT 920 Operating Manual*, 900MHz Spread Spectrum Industrial Ethernet Bridge, Rev 0.10, 19 October 2004.

10. Tucker, S., R. Vienneau, J. Corner, and R. Linderman, "Swathbuckler: HPC Processing & Information Exploitation." *Proceedings of the 2006 IEEE Radar Conference*, pp. 710–717, April 2006.

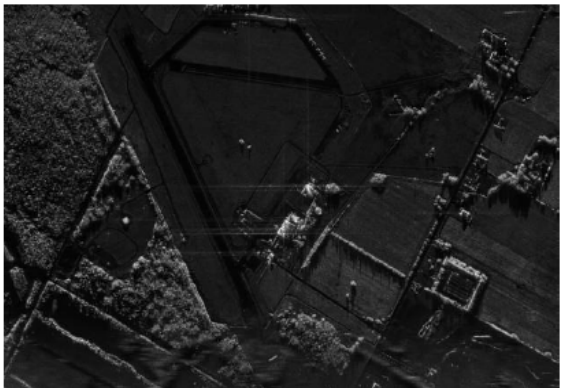

Figure 1. Sample Stripmap SAR image from AFRL HPC off-line processing

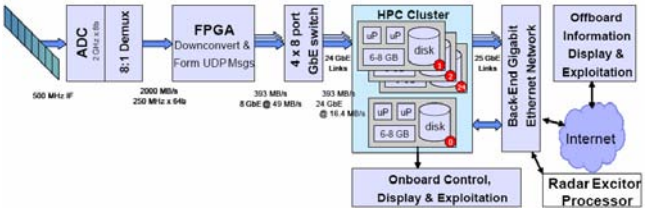

Figure 2. Swathbuckler system architecture with embedded HPC cluster

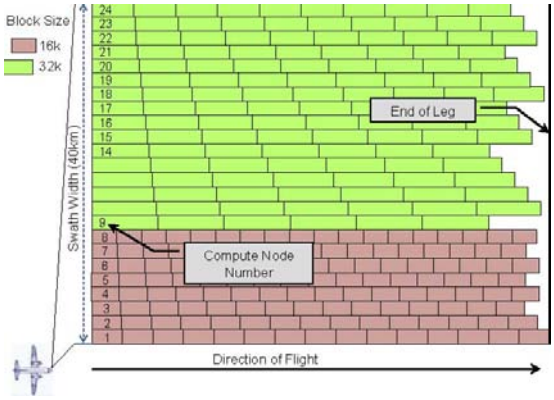

Figure 3. Illustration of data decomposition of SAR algorithm

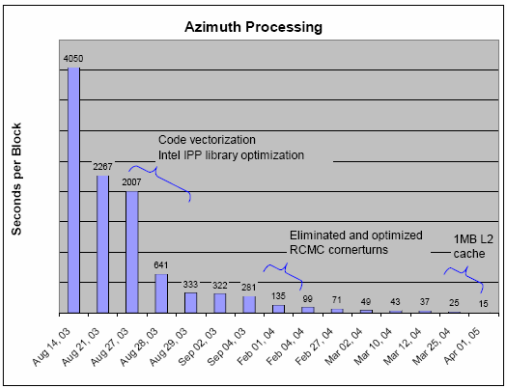

Figure 4. Azimuth processing reduction timeline

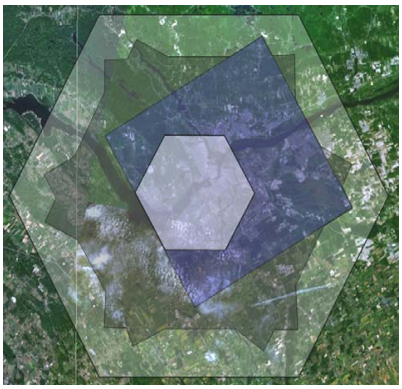

Figure 5. Ottawa, Ontario hexagon flight patter